```
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
from matplotlib import style
import pandas as pd
import pandas_datareader.data as web
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
/usr/local/lib/python3.6/dist-packages/pandas_datareader/compat/__init__.py:7: FutureWar
  from pandas.util.testing import assert_frame_equal
```

```
#getting data from yahoo  storing it in a csv and then reading it as a pandas dataframe
start = dt.datetime(2000,1,1)
end = dt.datetime(2019,12,31)
df = web.DataReader('AAPL', 'yahoo', start,end)
df.to_csv('apple.csv')
df = pd.read_csv('apple.csv',parse_dates = True, index_col =0)
print(df.shape) #5031 trading days are in this dataset
df.head()
```

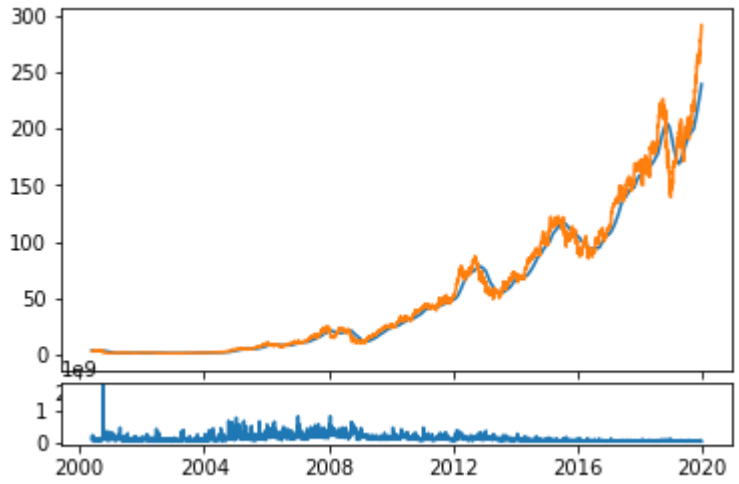
```
(5031, 6)
```

Date	High	Low	Open	Close	Volume	Adj Close
2000-01-03	4.017857	3.631696	3.745536	3.997768	133949200.0	3.460857
2000-01-04	3.950893	3.613839	3.866071	3.660714	128094400.0	3.169071
2000-01-05	3.948661	3.678571	3.705357	3.714286	194580400.0	3.215448
2000-01-06	3.821429	3.392857	3.790179	3.392857	191993200.0	2.937188
2000-01-07	3.607143	3.410714	3.446429	3.553571	115183600.0	3.076317

```
# getting moving averages of 100 days
df['100 days Moving Average'] = df['Adj Close'].rolling(window=100).mean()
df.dropna(inplace=True)
df.head()
```

Date	High	Low	Open	Close	Volume	Adj Close	100 days Moving Average
2000-05-	3.205357	2.964286	3.078125	3.131696	169615600.0	2.711102	3.545384

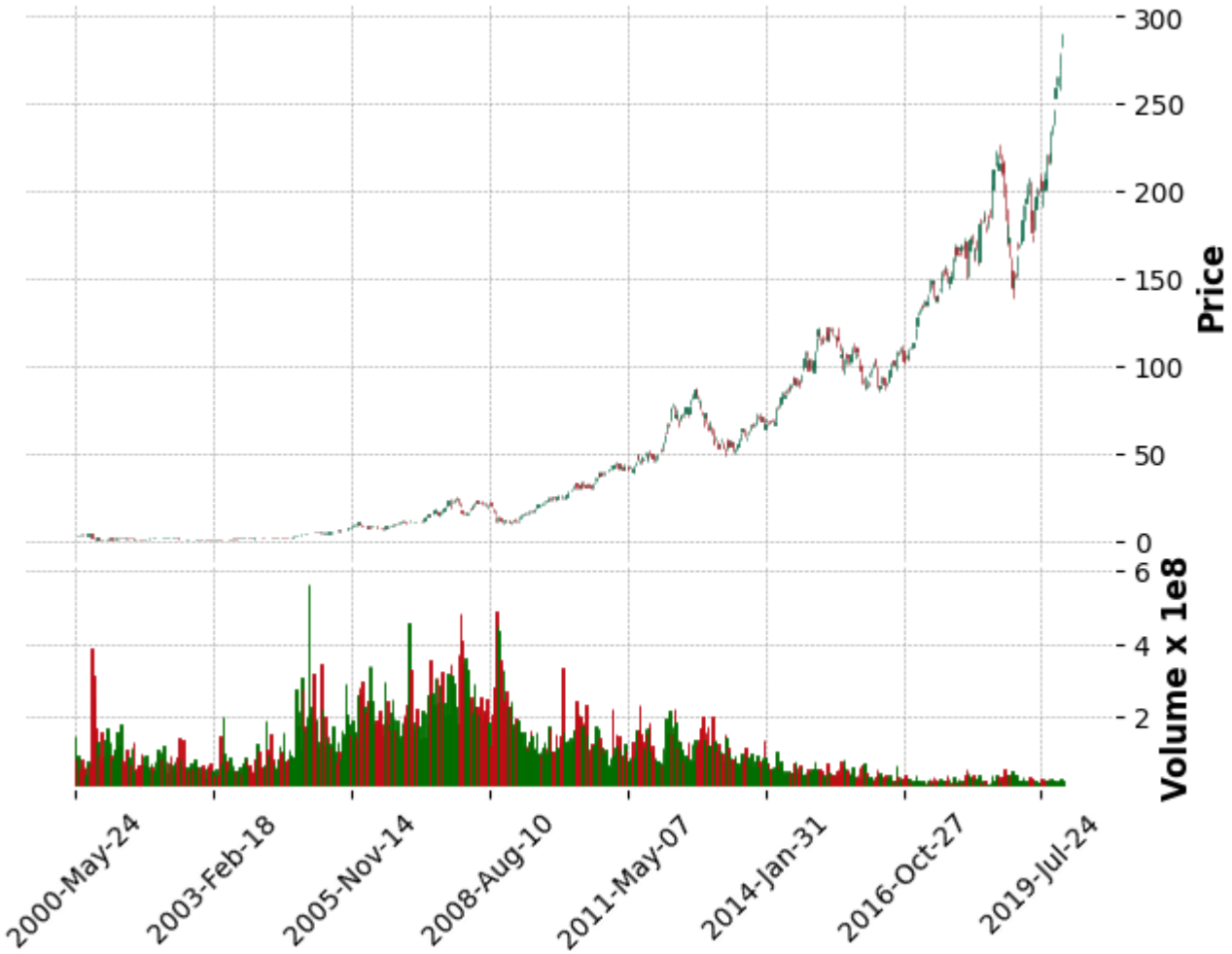
```
# visualizing the moving averages using subplots which are called axes
ax1 = plt.subplot2grid((6,1), (0,0), rowspan=5, colspan=1) #(6,1) -> 6 rows, 1 column (grid s
ax2 = plt.subplot2grid((6,1), (5,0), rowspan=1, colspan=1, sharex=ax1) #sharex actually makes

ax1.plot(df.index, df['100 days Moving Average'])
ax1.plot(df.index, df['Adj Close'])
ax2.plot(df.index,df['Volume'])
plt.show()
```

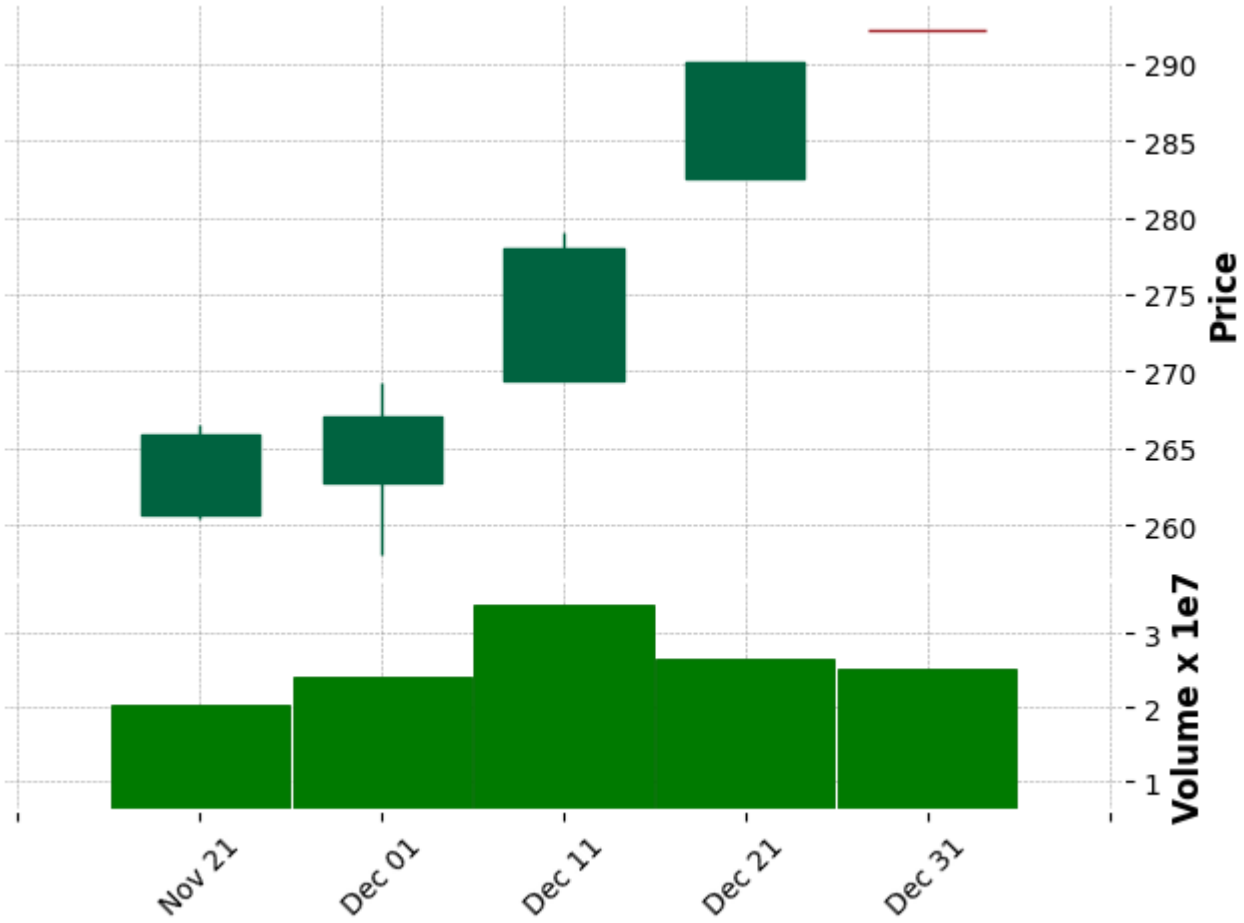



```
# creating a resampled dataset, the resample function is used to resample timeseries data
df_resampled = df['Adj Close'].resample('10D').ohlc() # ohlc does not sum or mean the values
df_resampled['Volume'] = df['Volume'].resample('10D').mean()
df_resampled.columns=['Open', 'High','Low','Close','Volume']
print(df_resampled.head())
print(df_resampled.shape) # condensed to 717 entries.
```

```
                Open      High       Low     Close        Volume
Date
2000-05-24  2.711102  2.861825  2.597093  2.861825  1.467720e+08
2000-06-03  2.823178  2.985497  2.819314  2.819314  8.197467e+07
2000-06-13  2.921729  3.439602  2.796125  3.323659  9.481710e+07
2000-06-23  3.196123  3.366172  3.169071  3.238636  5.862407e+07
2000-07-03  3.296607  3.640566  3.192259  3.640566  6.751140e+07
(717, 5)
```

```
import mplfinance as mpf
mpf.plot(df_resampled, type = 'candle', style='charles', volume=True)
mpf.plot(df_resampled.tail(), type = 'candle', style='charles', volume=True) # to show the zo
```


300
250
200
150
100
50
0
Price
6
4
2
Volume x 1e8
2000-May-24
2003-Feb-18
2005-Nov-14
2008-Aug-10
2011-May-07
2014-Jan-31
2016-Oct-27
2019-Jul-24


290
285
280
275
270
265
260
Price
3
2
1
Volume x 1e7
Nov 21
Dec 01
Dec 11
Dec 21
Dec 31

```
# Going to use original data (df) to create a target array
# target array will have predictions for what the close price will be in 100 days
df = pd.read_csv('apple.csv')
print(df.shape)
days = 50
df['Predicted_actual'] = df['Adj Close'].shift(-days) # predicted has the the 51st value from
print(df.head())
#df.dropna(inplace=True) # to remove the
```

```
(5031, 7)
         Date      High       Low  ...       Volume  Adj Close  Predicted_actual
0  2000-01-03  4.017857  3.631696  ...  133949200.0   3.460857          3.594190
1  2000-01-04  3.950893  3.613839  ...  128094400.0   3.169071          3.758441
2  2000-01-05  3.948661  3.678571  ...  194580400.0   3.215448          3.864721
3  2000-01-06  3.821429  3.392857  ...  191993200.0   2.937188          3.802885
4  2000-01-07  3.607143  3.410714  ...  115183600.0   3.076317          4.171965

[5 rows x 8 columns]
```

```
# Creating a feature matrix for sklearn
X = np.array(df.drop(['Date','Predicted_actual'],1))[:-days] #the last 50 values are removed
print(X)
# create target array
y = np.array(df['Predicted_actual'])[:-days]
print(y)
```

```
[[4.01785707e+00 3.63169646e+00 3.74553561e+00 3.99776793e+00
  1.33949200e+08 3.46085739e+00]
 [3.95089293e+00 3.61383939e+00 3.86607146e+00 3.66071439e+00
  1.28094400e+08 3.16907120e+00]
 [3.94866061e+00 3.67857146e+00 3.70535707e+00 3.71428561e+00
  1.94580400e+08 3.21544814e+00]
 ...
 [2.35240005e+02 2.33199997e+02 2.33369995e+02 2.34369995e+02
  1.84758000e+07 2.32485870e+02]
 [2.36149994e+02 2.33520004e+02 2.35089996e+02 2.35279999e+02
  1.68963000e+07 2.33388565e+02]
 [2.37580002e+02 2.34289993e+02 2.34589996e+02 2.36410004e+02
  2.43584000e+07 2.34509491e+02]]
[  3.59419012   3.75844073   3.86472106 ... 288.33331299 290.0446167
 292.16381836]
```

```
# Splitting the data into 20% testing and 80% training
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size = 0.2) #we use xtrain and ytra
```

```
# Decision Tree regressor is a tree based training algorithm
tree = DecisionTreeRegressor().fit(x_train, y_train)
tree.score(x_test, y_test) # this gets the model accuracy
```

```
0.963214934989716
```

```
#using the last 50 values of feature matrix to graph it against predictions made using the pr
#We graph them alongside actual values from the dataset.
#The last 50 values of the feature matrix would make predictions on the last 50 days of the a
feature_matrix =  df.drop(['Date','Predicted_actual'],1)[:-days]
feature_last_20 = np.array(feature_matrix.tail(days))
feature_last_20[:5]
```

```
array([[2.02759995e+02, 1.99289993e+02, 2.01300003e+02, 2.00990005e+02,
        2.46197000e+07, 1.99374222e+02],
       [2.02050003e+02, 1.99149994e+02, 1.99619995e+02, 2.00479996e+02,
        2.24749000e+07, 1.98868317e+02],
       [2.12139999e+02, 2.00479996e+02, 2.01020004e+02, 2.08970001e+02,
        4.72185000e+07, 2.07290085e+02],
       [2.06440002e+02, 2.02589996e+02, 2.03160004e+02, 2.02750000e+02,
        3.65474000e+07, 2.01120071e+02],
       [2.05139999e+02, 1.99669998e+02, 2.03460007e+02, 2.01740005e+02,
        2.72274000e+07, 2.00118195e+02]])
```

```
prediction = tree.predict(feature_last_20)
print(prediction) # these are the predicted values of the closing price of the last 50 tradin
```

```
[238.57650757 238.03094482 234.31108093 241.62182617 244.59771729
 247.04786682 241.33415222 241.30439758 246.76019287 253.76344299
 255.42993164 255.06292725 255.17201233 215.93003845 258.82345581
 260.87301636 260.63421631 263.13153076 261.31079102 253.76344299
 265.74822998 260.41534424 261.85803223 260.68399048 260.45513916
 265.02191162 264.23590088 165.15582275 165.15582275 262.8230896
 258.13696289 260.41534424 264.23590088 265.02191162 265.56912231
 267.12124634 269.3996582  153.13842773 151.8739624  278.60281372
 278.99087524 269.3996582  278.60281372 278.99087524 282.56268311
 282.83129883 288.44277954 290.0446167  290.0446167  292.16381836]
```

```
# Visualizing our predictions
Actual_last_20 = df.drop(['Predicted_actual','Volume','High','Low','Close','Open'],1)
Actual_last_20 = Actual_last_20[X.shape[0]:]
Actual_last_20['Prediction'] = prediction
last_100 = Actual_last_20.tail(100)
last_100_2 = df.tail(100)
plt.figure(figsize=(16,8))
plt.plot(last_100_2['Adj Close'])
plt.plot(last_100[['Adj Close','Prediction']])
plt.title('Predictive Analysis')
plt.xlabel('Days')
plt.ylabel('Price')
plt.legend(['Original', 'Actual','Predicted'])
```

```
<matplotlib.legend.Legend at 0x7fe105dca5c0>
```